Ampio Designer 2.0 - key changes

Document number: PO-265-EN Version: 1.0 Date of publication: September 14, 2026

Module firmware version **2.0 and newer** is released first for the **latest PCB versions**. Availability of the update for older hardware depends on the specific module and does not cover all devices.
The Ampio Designer interface changes and the new method of updating server services, described in the first part of this guide, work regardless of the hardware version.

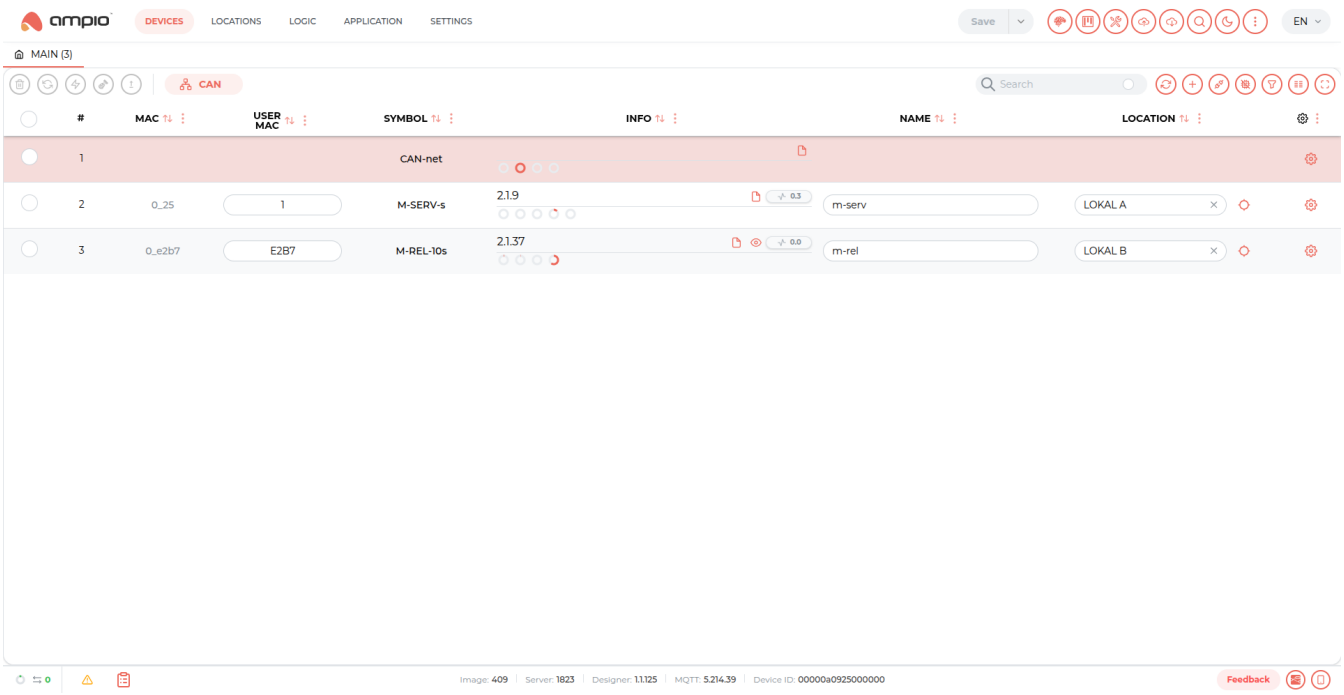
Introduction

Update 2.0 marks the beginning of major changes in Ampio Designer, aimed at improving day-to-day work with the system. The current stable release brings primarily a **completely refreshed interface** and **new diagnostic tools**, such as CAN bus load statistics.

A much wider range of new features, covering module communication and firmware, logic creation, events and grouping, is tied to updating modules to version 2.0 and remains under active development and testing (see the notice below).

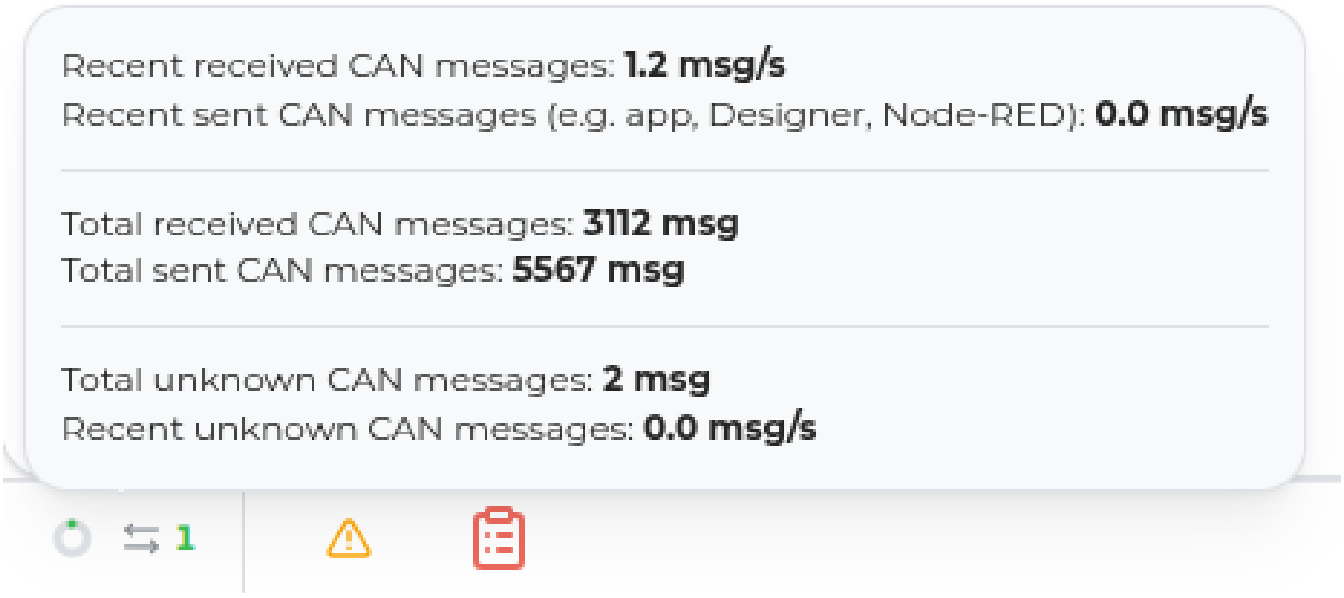
Visual and interface changes

Ampio Designer has gained a new look, and the added functions shorten the configuration time of repetitive elements while making it easier to diagnose and supervise the installation.

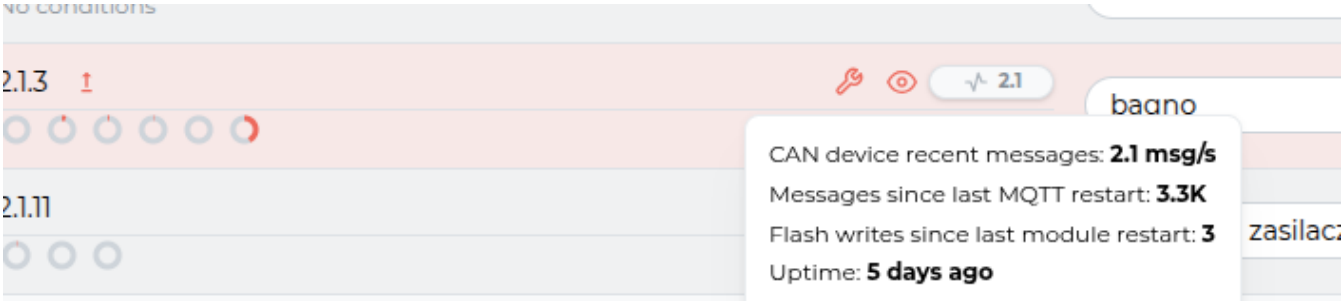


CAN bus statistics

A new indicator on the bottom bar displays the current bus throughput, showing how heavily the system is loaded. This tool is useful for diagnosing problems, for example when logic created in the Node-RED environment unexpectedly causes a sharp increase in the number of messages, which may affect the stability of the system.



The same statistics are also available for individual modules in a condensed form, which makes it possible to quickly identify the devices generating the most network traffic.



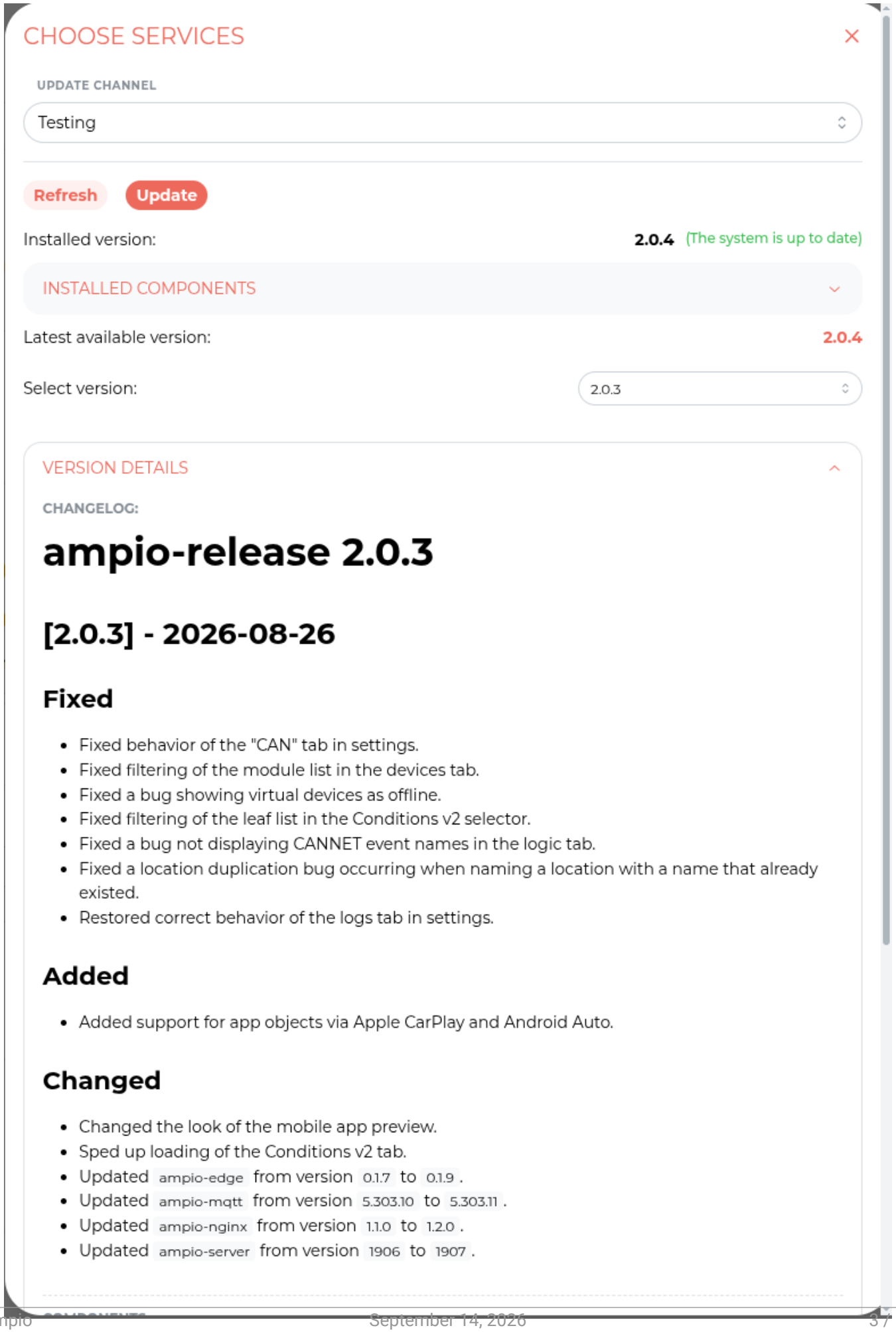
The condensed module statistics include:

- **CAN device recent messages:** the current traffic intensity generated by the module (msg/s).
- **Messages since last MQTT restart:** a cumulative counter, useful for assessing traffic over a longer period.
- **Flash writes since last module restart:** information about the number of write operations to the device memory.
- **Uptime:** how long the module has been running since its last restart.

Server service updates (Ampio APT)

The way server software is updated has been completely rebuilt. Previously each software package was updated separately. This required manually keeping versions consistent between components and gave no clear information about which set was the stable version and which was the test one.

The system is now updated as **one coherent release version** (*ampio-release*), containing a complete set of components tested together.


CHOOSE SERVICES
UPDATE CHANNEL
Testing
Refresh
Update
Installed version:
2.0.4 (The system is up to date)
INSTALLED COMPONENTS
Latest available version:
2.0.4
Select version:
2.0.3
VERSION DETAILS
CHANGELOG:
ampio-release 2.0.3
[2.0.3] - 2026-08-26
Fixed
Fixed behavior of the "CAN" tab in settings.
Fixed filtering of the module list in the devices tab.
Fixed a bug showing virtual devices as offline.
Fixed filtering of the leaf list in the Conditions v2 selector.
Fixed a bug not displaying CANNET event names in the logic tab.
Fixed a location duplication bug occurring when naming a location with a name that already existed.
Restored correct behavior of the logs tab in settings.
Added
Added support for app objects via Apple CarPlay and Android Auto.
Changed
Changed the look of the mobile app preview.
Sped up loading of the Conditions v2 tab.
Updated ampio-edge from version 0.1.7 to 0.1.9 .
Updated ampio-mqtt from version 5.303.10 to 5.303.11 .
Updated ampio-nginx from version 1.1.0 to 1.2.0 .
Updated ampio-server from version 1906 to 1907 .

The new update window provides:

- **Update channel:** a choice between the **Stable** and **Testing** channels, shared across the whole system.
- **Version selection:** the stable channel offers the current stable version, while the test channel shows subsequent releases, including stable ones. The release history is built from the moment the new update system was introduced, so initially it covers only the most recent entries.
- **Installed components:** an expandable list of all packages included in the release, along with their versions.
- **Version details:** the changelog of the selected release, divided into fixes, new features and changes, so that before updating you know exactly what is going to change.

The **Testing** channel is intended for familiarising yourself with upcoming features on test installations. On end-customer installations we recommend using the **Stable** channel exclusively.

Features in the beta phase

Most of the new features described below are tied to **updating modules to version 2.0**. This version is currently **under active development and intensive testing**.
We are making it available so that you can familiarise yourself with the upcoming capabilities early, however **using it is entirely at your own risk**:

- We **do not recommend** testing version 2.0 on **end-customer installations**. Until the stable release, use it exclusively on test installations.
- The features and the way they are configured **may change** in subsequent releases.
- Some of the features described below may still be **unavailable in the current release** and will appear with future updates.

To use Ampio Designer in version 2.0 the following is **required**:

- updating the server software to release **ampio-release 2.0.1 or newer**
- updating the **modules** to version **2.1 or newer**
- having modules in a hardware version that supports firmware 2.0 (see the notice at the beginning of this guide)

All server services, including *Ampio Designer*, *Ampio MQTT* and *Ampio Server*, are part of a single release, so there is no need to verify their versions separately. The update procedure is described earlier, in the *Server service updates (Ampio APT)* section.
To ensure that all functionalities are available, we recommend always using the current stable versions.

In Ampio Designer 2.0 we introduce the concept of a **LEAF**. It is the elementary component of the system. It can be a physical input/output, but also a flag, a group or an event.

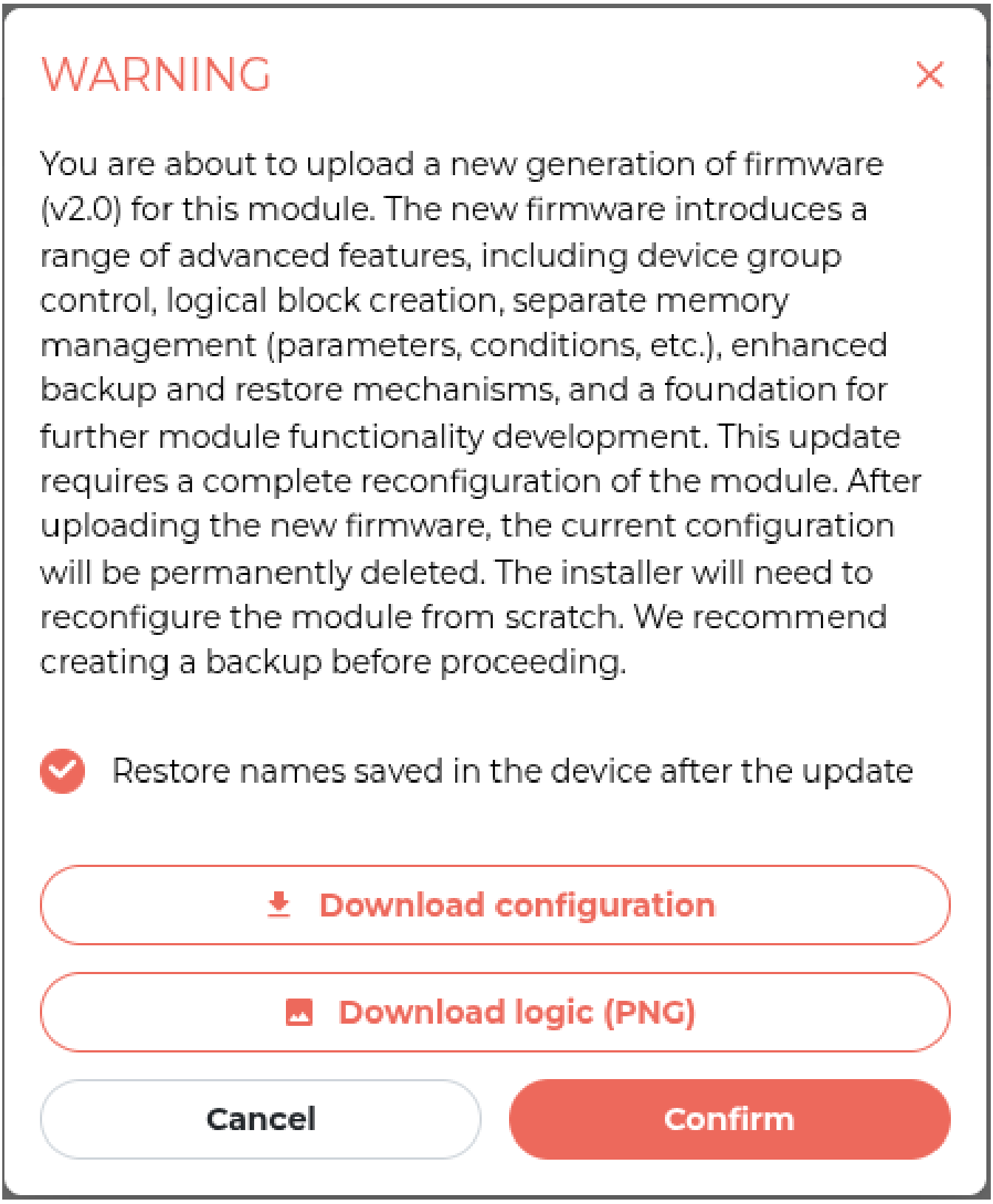
Updating the system and the modules

Updating the server software and updating the modules are two **independent decisions**:

- **Server software:** an update intended for all installations. It introduces the interface changes described earlier, the diagnostic tools and the new way of updating services.
- **Module firmware:** a decision made individually for a given installation. It is worth considering when the new features described in this part of the guide are needed, while accepting that the logic will have to be created again.

To take full advantage of the new functionalities, the modules must be updated to version **2.1 or newer**.

Before the update is performed, a message is displayed informing about its consequences, that is about a **partial removal of the configuration**.
Custom names stored on the module are preserved as long as the **Restore names saved in the device after the update** option stays selected. The **conditions are deleted**, which is why the same window offers the **Download configuration** and **Download logic (PNG)** buttons, the latter being a file with a graphical representation of the conditions previously stored on the module (with a larger number of conditions this is a ZIP archive with several images), which makes it easier to recreate them in the new system. The files are not downloaded automatically, the installer decides before confirming the update.



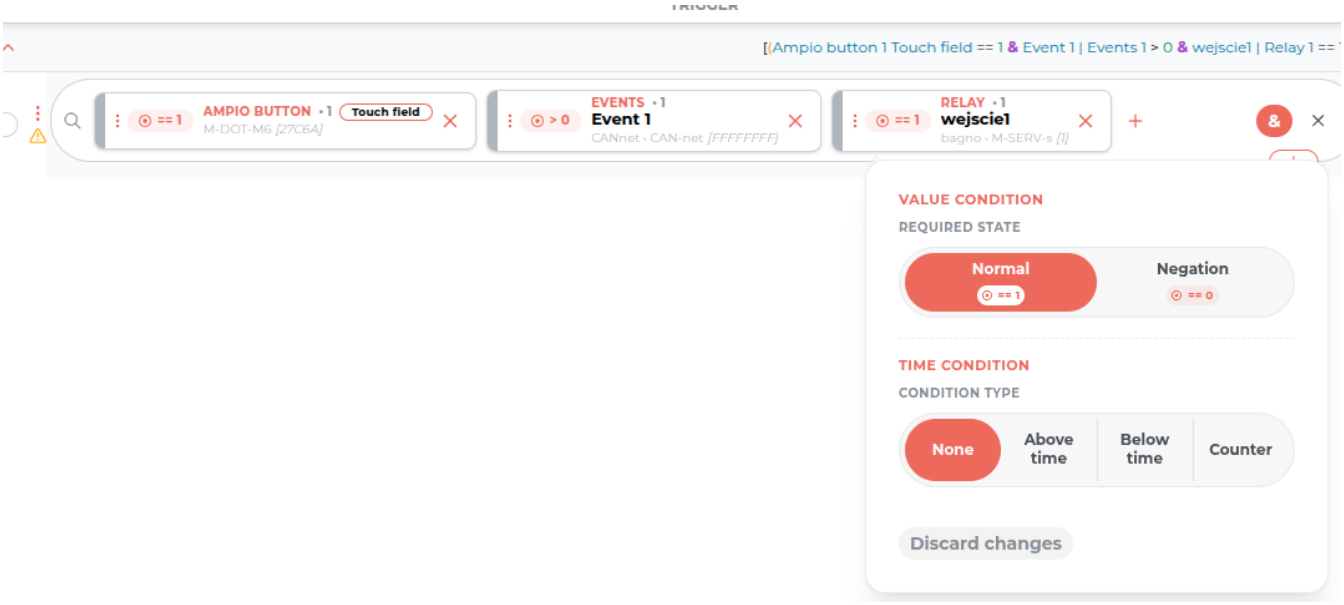
Logic and conditions

The changes described below apply only to **conditions in version 2.0** and require modules with firmware version **2.1 or newer**.

The new structure of conditions

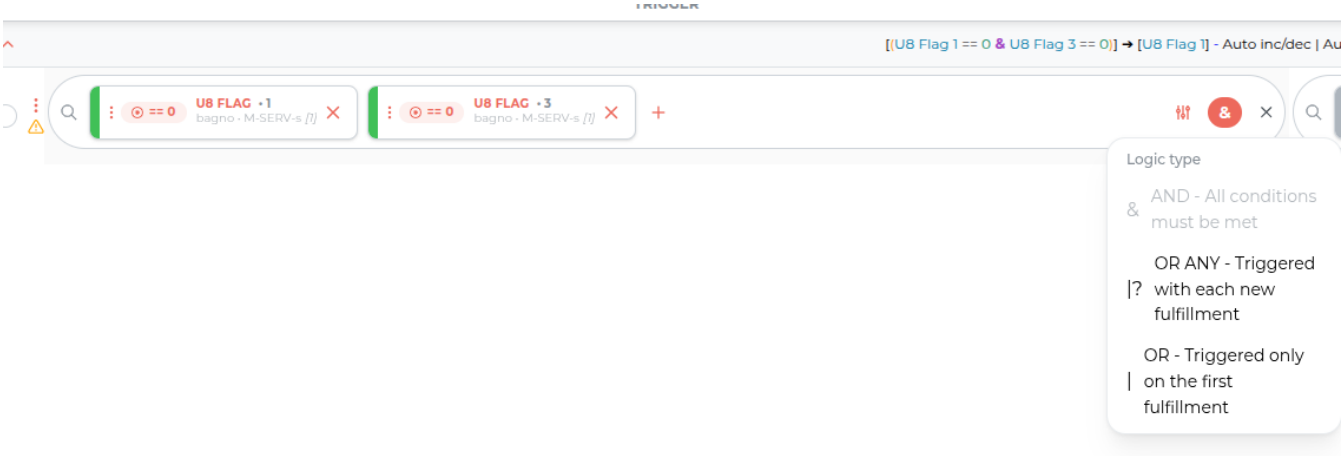
Conditions in version 2.0 have been rebuilt to make it possible to create complex logic.

1. Triggers A single trigger can now combine Leafs originating from different modules. To configure the parameters of a given trigger (for example the value or the duration required for activation), simply click the name of the leaf in the trigger.

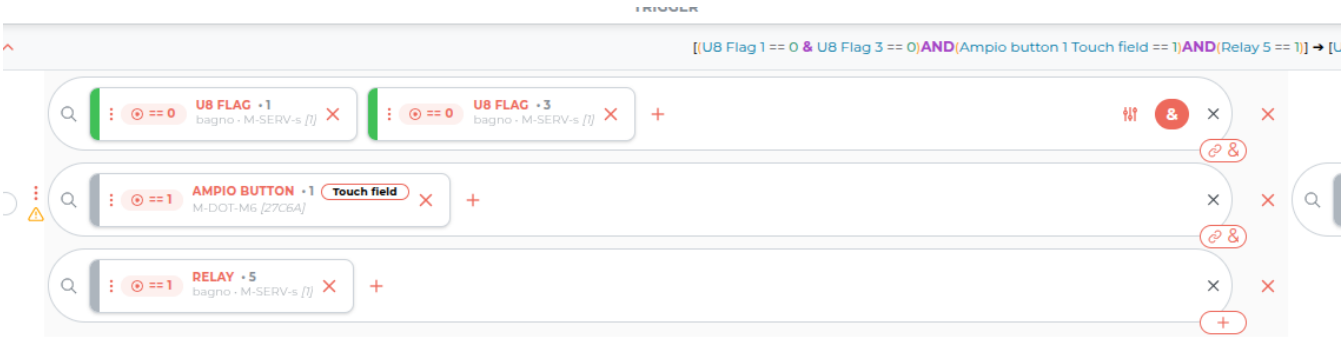


2. Logic inside a block Within a single trigger block you can now choose the logic type, which decides when the block is triggered:

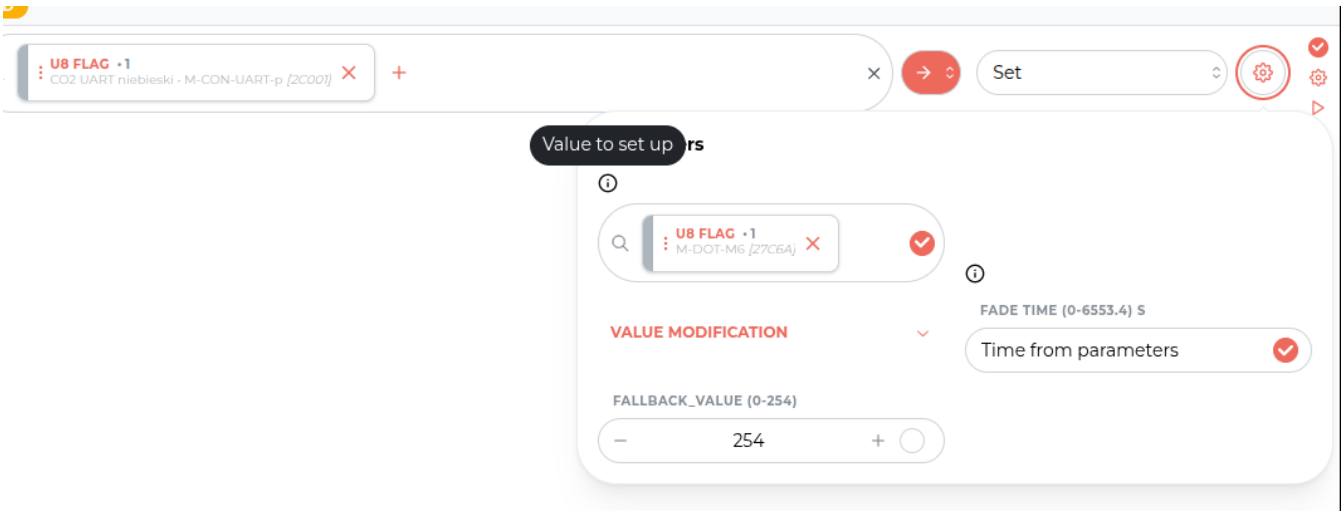
- **AND (&):** all conditions in the block must be met.
- **OR ANY (|?):** triggered with each new fulfilment of any condition in the block.
- **OR (|):** triggered only on the first fulfilment of the condition.



3. Trigger blocks Linked conditions have been replaced by **trigger blocks**. Up to **3 trigger blocks** can be created in total. The logic joining the individual blocks is always of the **AND** type. This means that in order to execute the action, the conditions defined in **every** block used must be fulfilled.



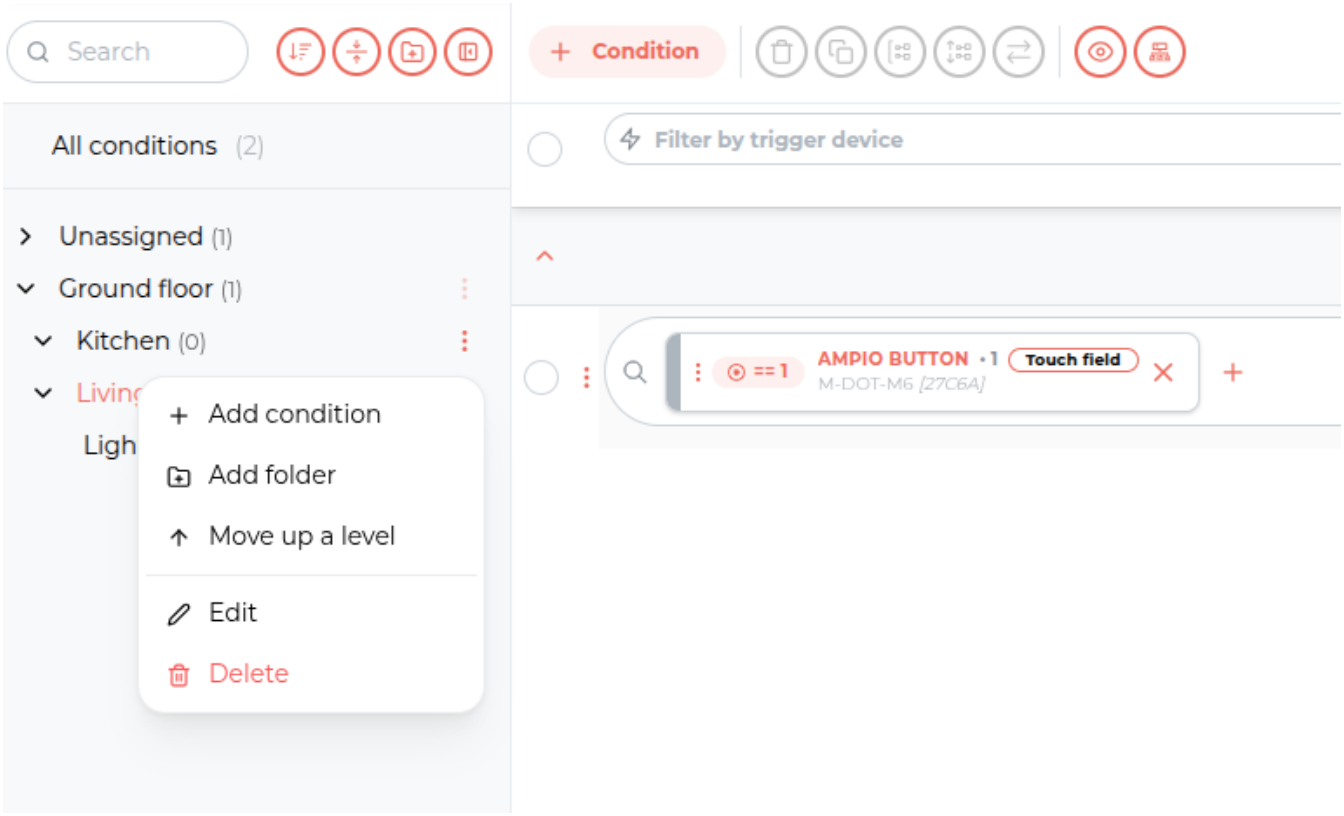
4. Value injection **Value injection** allows a value to be passed dynamically from one element (leaf) to another within the executed action. For example: a flag value set by the user in the Ampio UNI application can be “injected” as the target brightness level of the lighting in a condition.



Condition folders

A folder system has been introduced, replacing the previous system of grouping conditions and allowing the logic in a project to be organised. Folders work exactly like those in a file system:

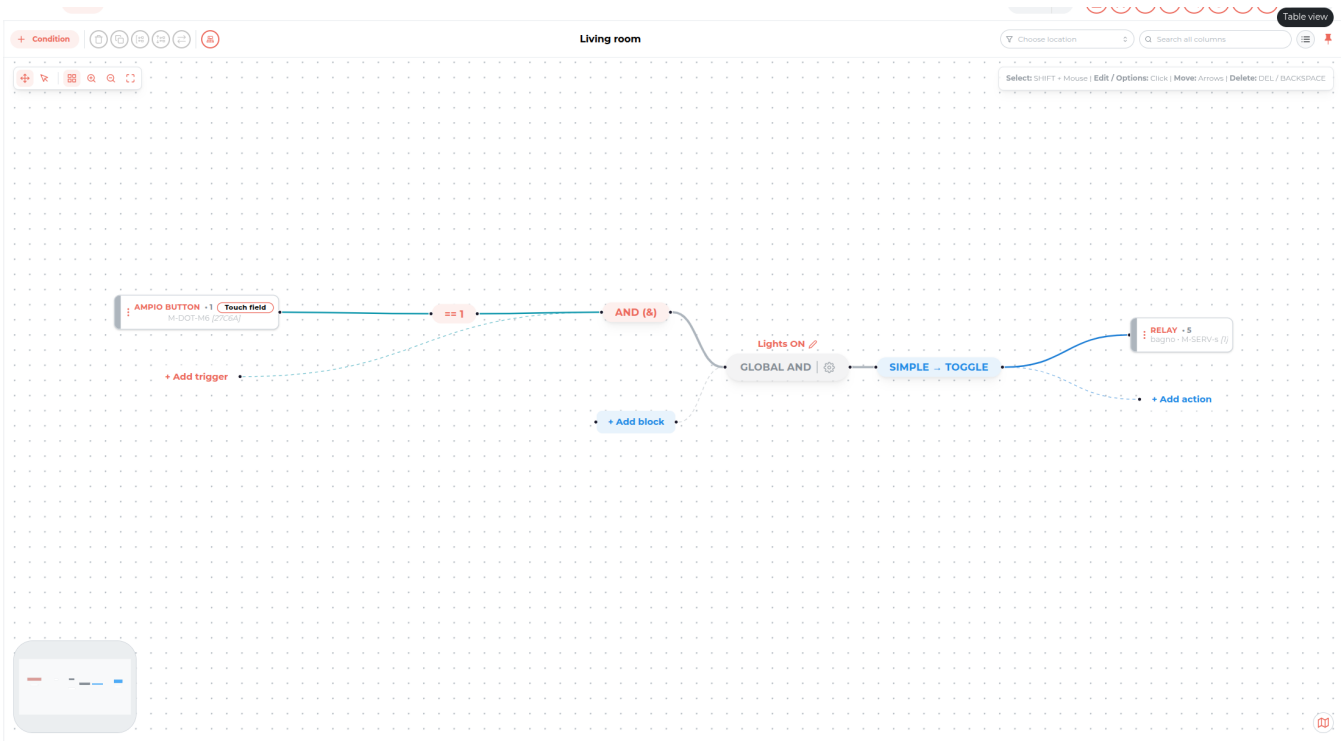
- You can create a folder structure corresponding to locations or functions.
- Conditions are moved between folders using **drag and drop**.
- The folder list can be hidden at any time to gain more space on the working screen.



Alternative block view of conditions

The logic tab offers a new way of presenting conditions, which allows the logic to be visualised as a graph. This changes the way relationships are analysed and created:

- **Flow visualisation:** Presents the entire structure of dependencies, from the trigger through the logic blocks to the actions, allowing the logic to be understood at a glance.
- **Diagnostics:** The block view acts as an interactive diagnostic tool. When a given part of the condition is fulfilled, the line connecting the blocks is set in motion (a flow animation). This makes it possible to verify immediately which element of the system activates the logic and which one is blocking it at that moment.
- **Space management:** The view supports free panning and scaling of the working area and of the blocks, which makes work easier with a large number of interconnected blocks.

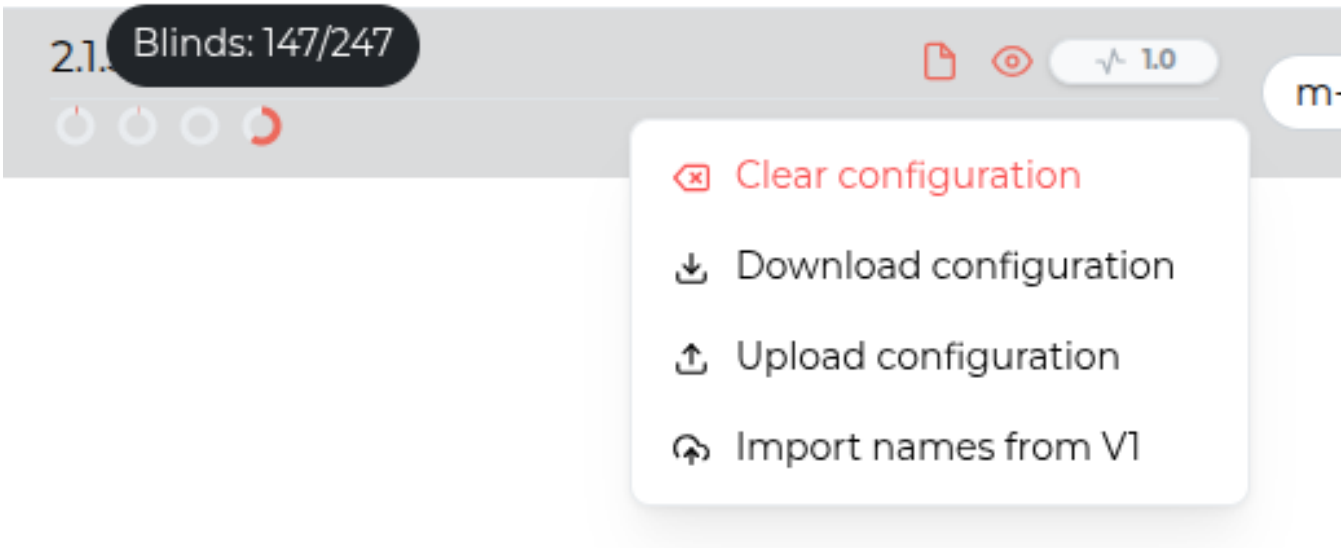


The block view is fully synchronised with the classic list view. Switching between them does not affect the logical structure of the condition, only the way it is presented and edited. Changes made on the graph are immediately reflected in the linear layout.

Modules and their firmware

Module sections

Below the current firmware version of a module, its **memory sections** are shown. This makes it possible to determine precisely how many resources are taken up by, for example, the current descriptions or the logic.



Section import/export

Between the custom name of the module and its firmware version there is a configuration menu. It allows you to:

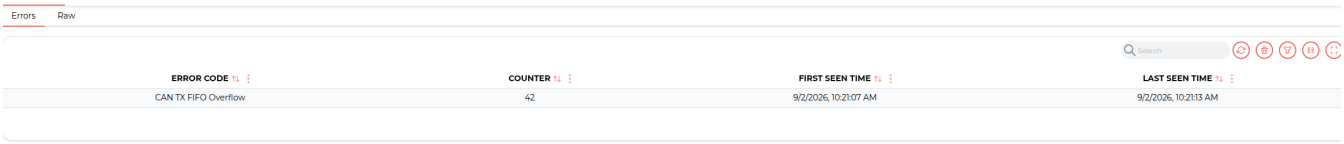
- **Import and export** individual module sections.
- **Clear the data**, that is, quickly restore the device to its factory configuration.

Error diagnostics

If a module has registered an error, an **ERROR** marker appears next to its statistics in the device list. Thanks to this the problem is visible immediately at the level of the whole installation, without having to review the devices one by one.

The details are available after entering a given module, in the **Errors** tab. For each error the following is presented:

- **Error code:** a description of the event, for example *CAN TX FIFO Overflow*.
- **Counter:** the number of occurrences of the given error.
- **Time of first occurrence** and **time of last occurrence:** these make it possible to determine whether the problem is momentary or recurring continuously.

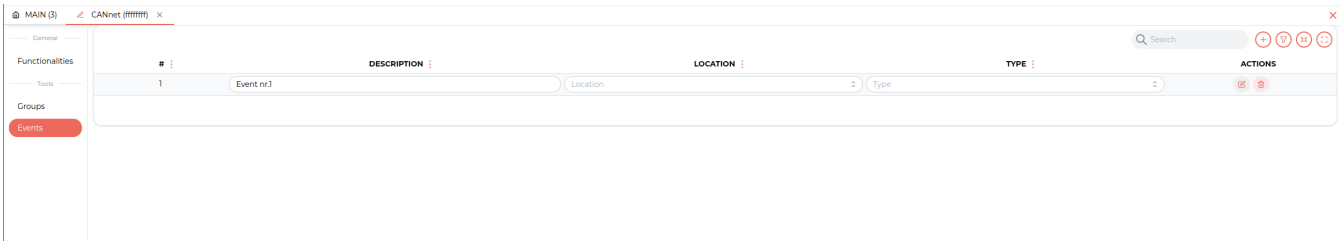


The list can be sorted, searched and filtered, and the counters can be cleared in order to verify whether the error still occurs after changes have been made to the installation. Next to it there is a **RAW** tab with a raw preview of the diagnostic data.

CAN-net

An additional virtual module named **CAN-net** is visible in the devices tab. It is responsible for storing the descriptions of conditions and condition groups. This module is also used to configure events and the newly added group functionality.

Events Event configuration has been moved from the global settings to the **CAN-net** virtual module. New events are defined in the module's tools tab. It is also now possible to trigger events manually directly from the functionalities tab, which allows the correctness of the logic to be verified efficiently.

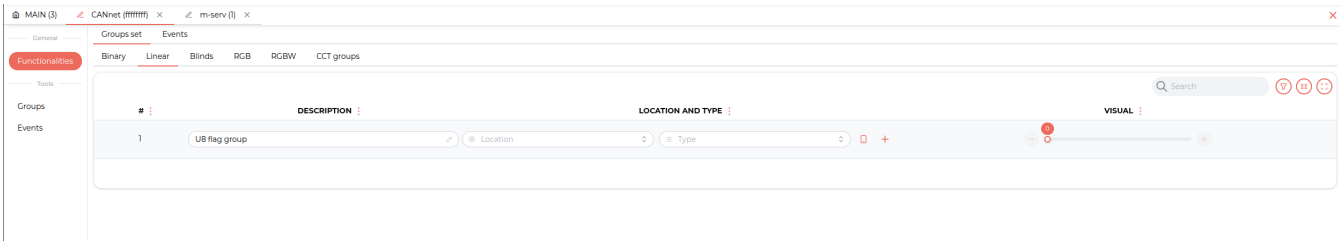


The key change in version 2.0 is the transition from a pulse-based model to a value-based one:

- **Previously:** An event acted only as a signal that a given action had occurred.
- **Now:** An event acts as a carrier of a value. While configuring the logic you can specify the particular value with which the event will be sent to the bus.

This approach allows a single event to be reused for different purposes. For example: one event, depending on the value assigned to it, can trigger different lighting scenes or control how far the blinds are opened.

Groups The **CAN-net** module introduces the functionality of grouping Leafs, configured from the module's tools. Groups make it possible to manage resources distributed across the system collectively.



The following group types are available:

- **Binary:** intended for elements with two states (ON/OFF), such as relay outputs, binary flags or the control of air-conditioning units via an HVAC module.
- **Linear:** dedicated to Leafs with linear values, for example U8 flags, dimmers or M-LED module channels.
- **Blinds:** provide integrated control of the blind position and of the slat tilt angle.
- **RGB:** allow collective control over the colour and brightness of lighting.
- **RGBW:** extended support for colour lighting with an additional white channel, allowing precise management of the colour or white saturation level.
- **CCT:** dedicated to managing lighting with a variable colour temperature.

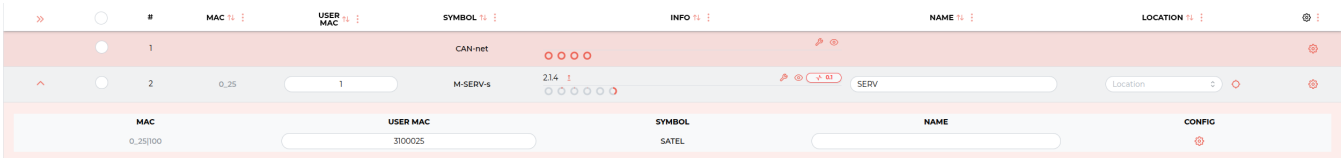
From the functionalities list of the **CAN-net** module it is possible to control the value of the created groups manually.

> Because of the universal character of groups, the set of actions available for them in conditions is simplified compared with the functions offered for individual Leafs. This allows many different Leafs to be controlled consistently and reliably within a single group.
> If advanced functions dedicated to a particular device type are required, we recommend controlling the individual Leafs directly within the condition.

Child modules

Modules handling communication with external devices were previously presented as a single device, and all elements coming from the integration ended up in a shared functionalities list of the parent module.

Version 2.0 introduces **child modules**. The parent module is still responsible for all communication, both physically and in software. In the device tree, each integration connected to it receives **its own separate device**, containing only its own elements.



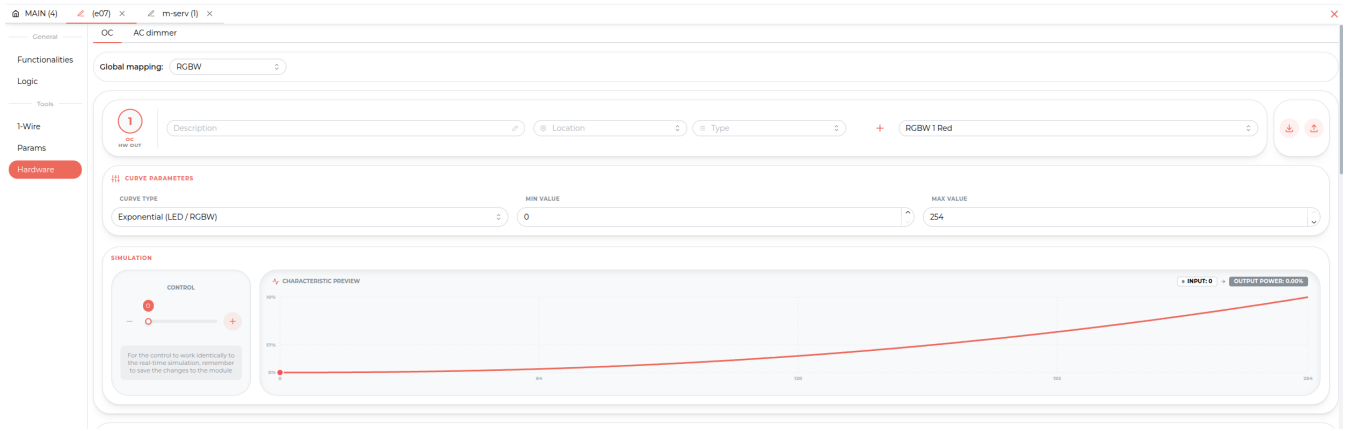
This mechanism covers:

- **Satel / RS-232:** the integration with the alarm control panel is visible as a separate child device.
- **RS-485:** each device polled over Modbus constitutes a separate child of the parent module.
- **LoRa:** the base station presents the devices connected to it as child modules.

As a result, Leafs coming from the individual integrations do not mix with each other or with the functions of the parent module itself, which simplifies finding them while creating logic and organises the structure of the project.

Mapping

In modules that support output mapping, a new **Hardware** tab has been introduced, organising the functions related to mapping and to the physical settings of the outputs.

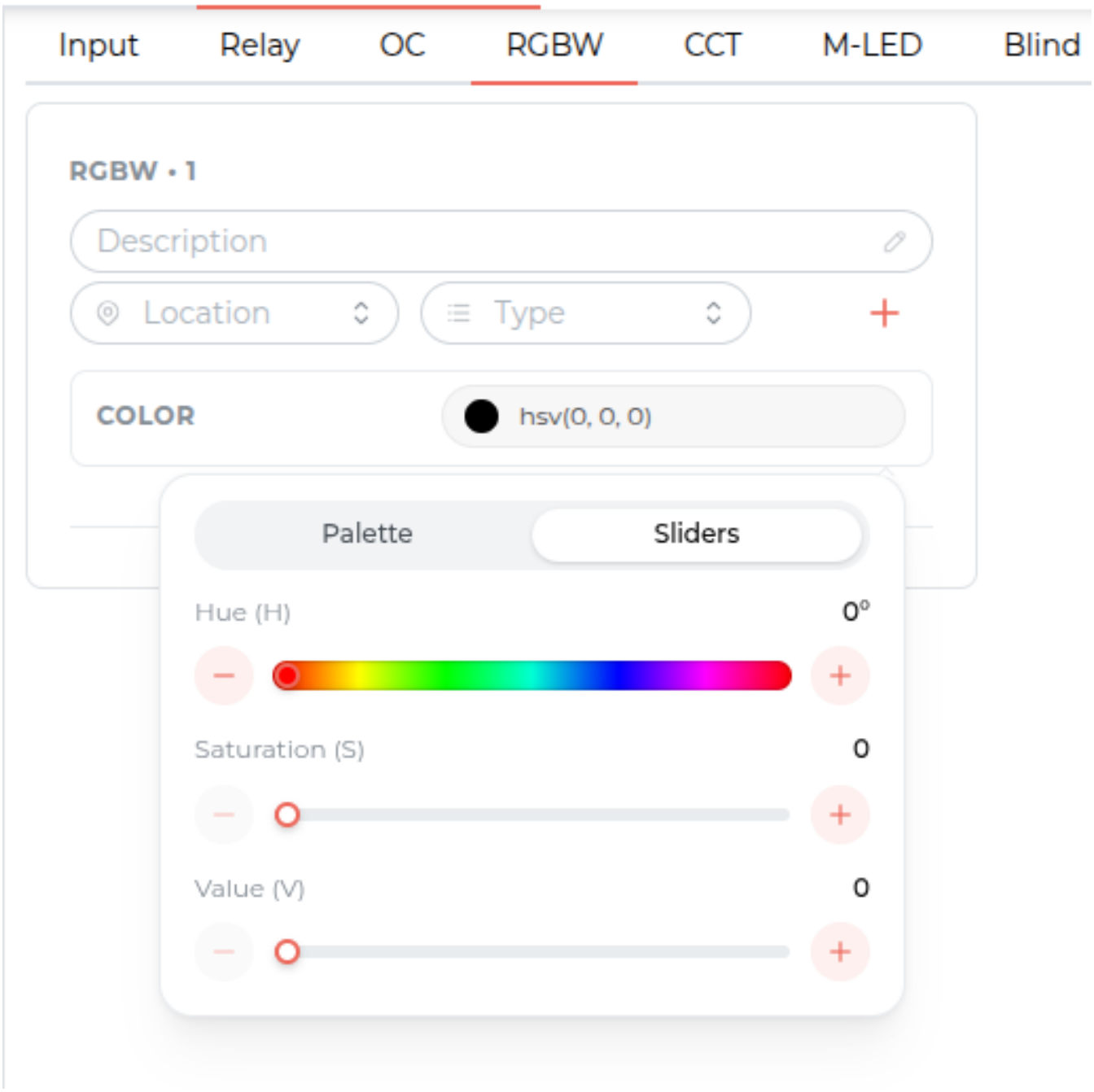


From this section it is possible to change the mapping of individual channels and to fine-tune their operating parameters, such as:

- minimum and maximum values,
- the choice of the control curve type.

RGBW control (HSV model)

The way RGBW colour lighting is managed has been completely rebuilt and based on the **HSV** standard (Hue, Saturation, Value). In contrast to classic control of the individual channels (RGB), this model allows much more intuitive management of light.



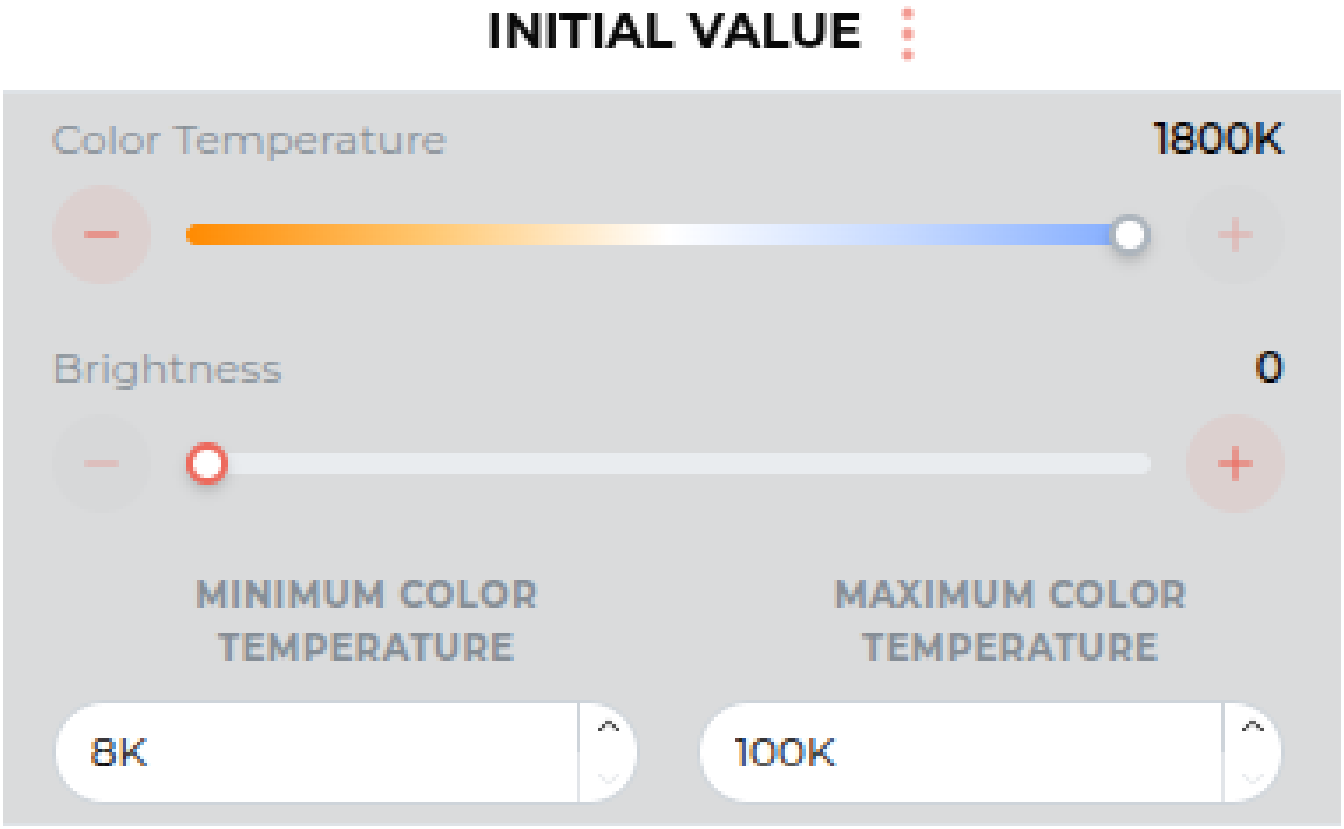
The new interface allows three parameters to be operated independently:

- **Hue (H):** selection of a particular colour from the full palette.
- **Saturation (S):** determines the intensity of the selected colour. Reducing the saturation smoothly adds the white channel (**W**), up to pure white.
- **Value (V):** smooth regulation of the light intensity for the whole leaf, without affecting the previously selected colour or saturation.

Modules equipped with 8 OC-type outputs (for example **M-OC-C8s** and **M-INOC-8s**) have gained the ability to handle **two independent RGBW channels** from firmware version **2.1 or newer**.

CCT control

Support for light sources with a variable colour temperature (CCT) has been rebuilt. The previous proportional control model has been replaced by a model based on physical parameters, which allows much greater precision.



The most important changes include:

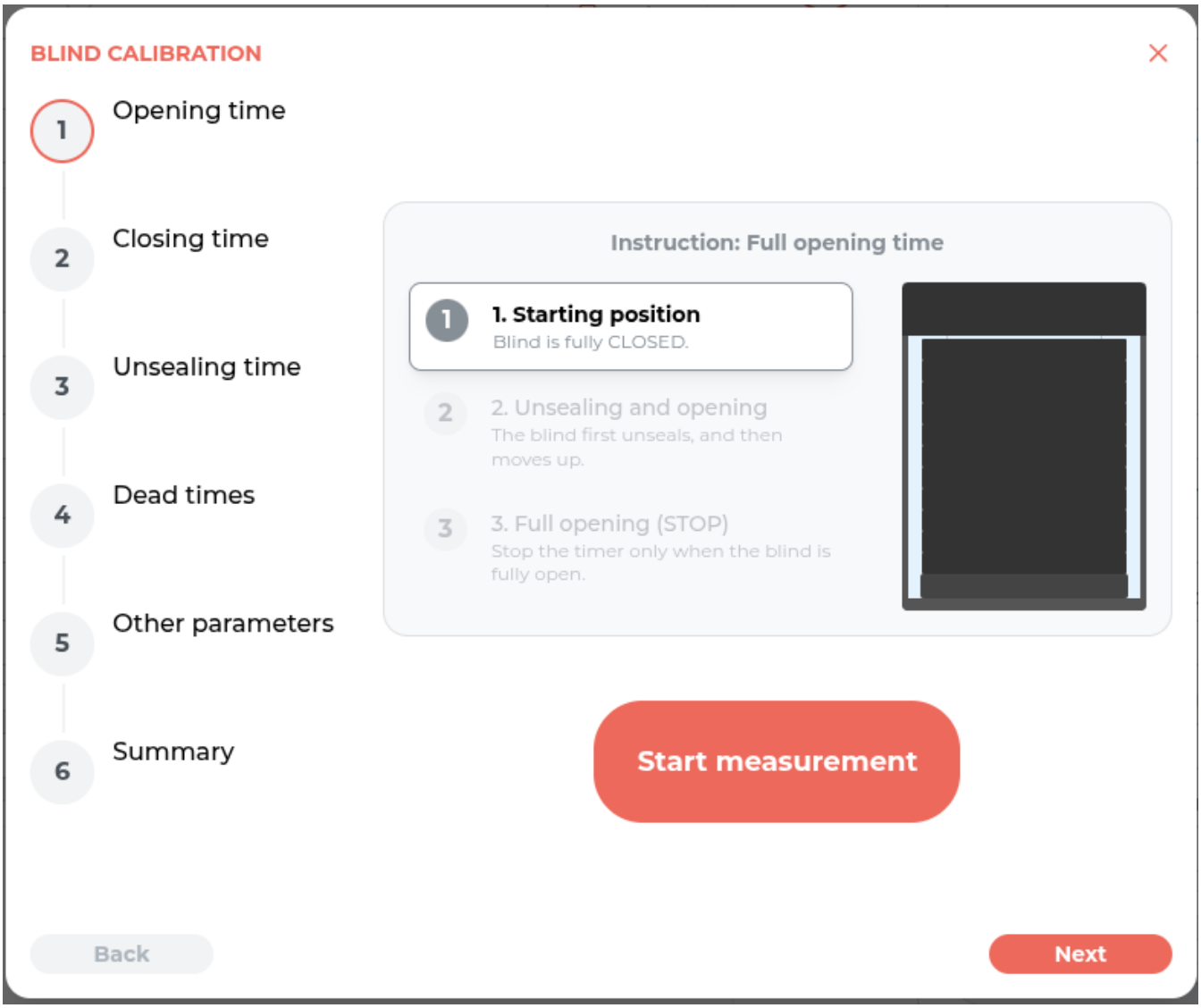
- **Working with physical values:** Instead of operating a proportional slider, the system allows work with the actual colour temperature expressed in kelvins (K).
- **Range definition (Min/Max):** In the parameters section it is possible to define the **minimum and maximum colour temperature** for a particular leaf.
- **Matching the hardware:** Thanks to the limited ranges, the control slider operates only within the limits supported by the given LED strip or luminaire, which prevents setting colours outside the specification of the light source.

The server now also supports mapping OC outputs as **CCT** channels.

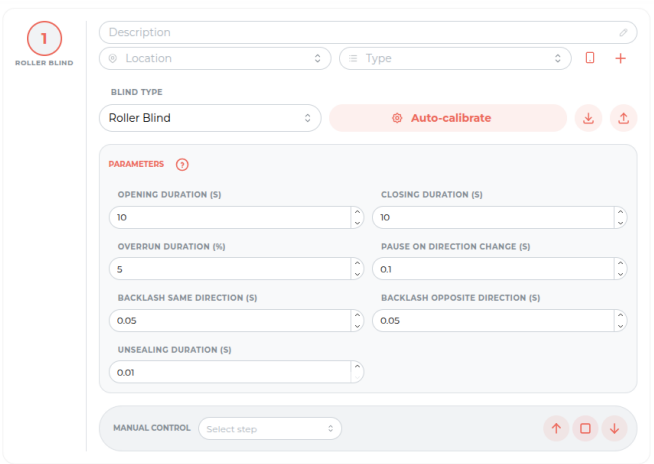
Blinds

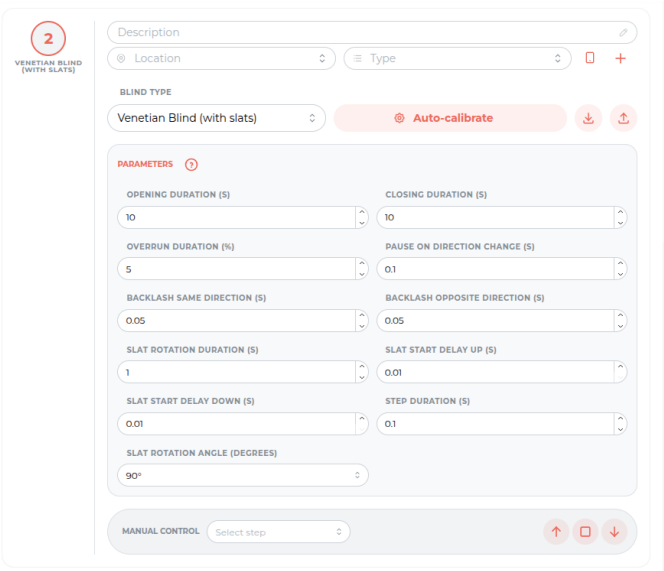
Support for blinds and venetian blinds has been extended with a wizard that measures the drive parameters automatically, and with a refreshed parameters panel.

Auto-calibration The **Auto-calibrate** button opens an interactive wizard that guides the installer step by step through the process of configuring the drive. On the basis of the measurement cycles performed, all the key operating parameters of the blind are determined: the opening and closing times, the unsealing time and the dead times.



New parameters interface The parameters panel has been completely refreshed.



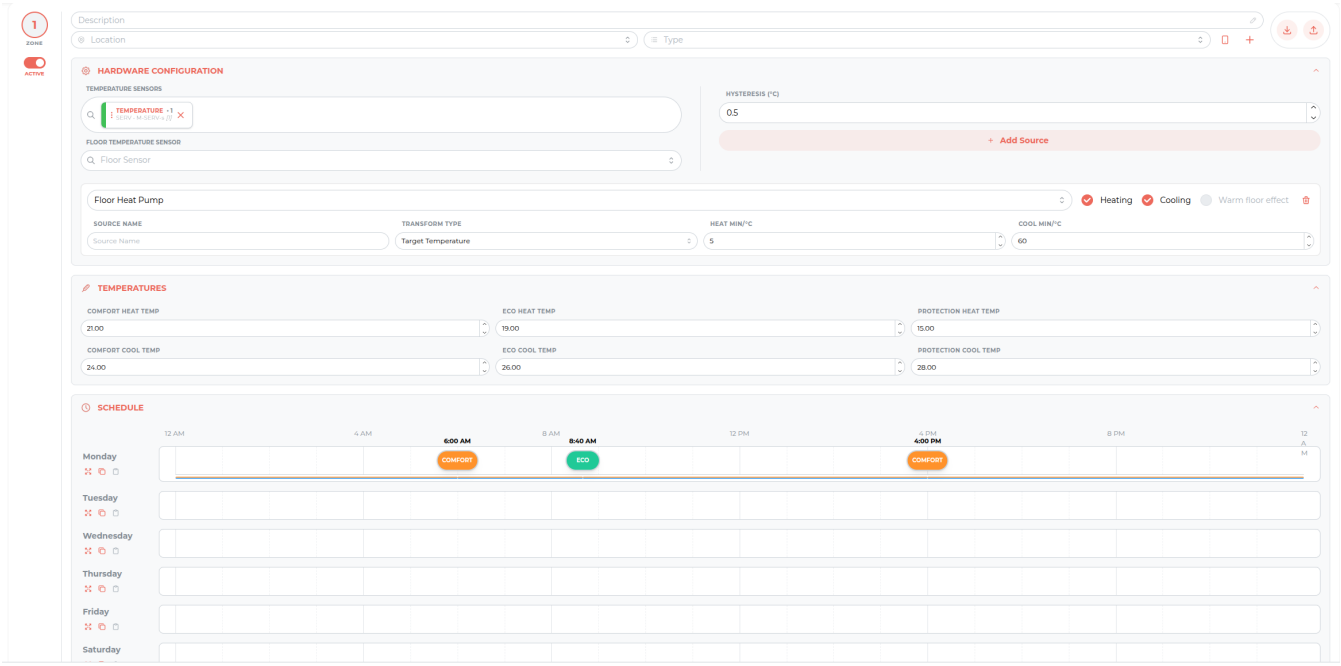


Key changes in the parameters:

- **New names for the existing settings:** The parameters known from the previous version work in the same way, but have been given clearer names:
  - *Operating mode: Standard / With slats* → **Blind type: Roller Blind / Venetian Blind (with slats)**
  - *Additional calibration (%)* → **Overrun duration (%)**
  - *Safe direction change delay (s)* → **Pause on direction change (s)**
  - *Motor start-up time - travel in the same direction (s)* → **Backlash same direction (s)**
  - *Motor start-up time - travel in the other direction (s)* → **Backlash opposite direction (s)**
- **Unsealing duration:** For roller blinds there is a separate **Unsealing duration (s)** parameter, also determined by auto-calibration.
- **More detailed slat parameters:** Previously the operation of the slats was described by a single *Slat travel time (s)* parameter. In the **Venetian Blind (with slats)** mode it has been replaced by a complete set of separate settings: **Slat rotation duration**, **Slat start delay up** and **Slat start delay down**, **Step duration** and **Slat rotation angle**. This makes it possible to eliminate inaccuracies resulting from the mechanical construction of the venetian blind itself.

Controller

Support for heating and cooling zones has been designed completely from scratch. The previous, relatively simple controller has been replaced by an extensive mechanism that makes it possible to describe the actual operation of a heating installation, with multiple heat sources, a schedule and separate handling of the floor temperature.



The most important new features:

- **Schedules in Ampio Designer:** The controller's operating schedules can now be configured directly from Ampio Designer.
- **Multiple sources:** The ability to define different heating and cooling sources for a single zone. For example, the schedule can indicate which source, for example an HVAC module, takes part in reaching the setpoint temperature in a given time interval.
- **Sensor integration:** The new interface allows the operation of the controller to be linked instantly with presence sensors or window reed switches.

Underfloor heating Dedicated support for underfloor heating has been introduced, based on a separate **floor sensor**. Thanks to this the controller can simultaneously supervise two quantities: the air temperature in the room and the temperature of the floor itself.

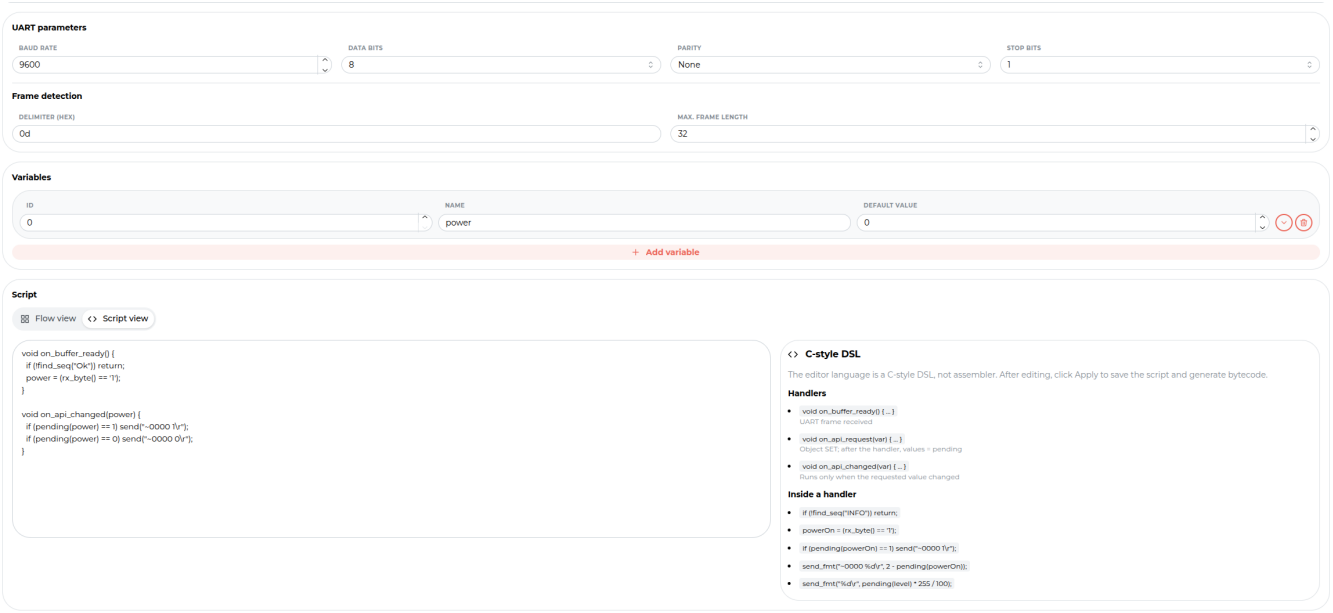
This solution makes it possible to implement the **warm floor effect**. The floor is kept in a state of constant, minimal heating in order to preserve comfort when walking on it, regardless of whether the room has already reached the setpoint air temperature.

Operation in this mode is covered by **safety limits** which prevent the floor from overheating. This protects both the floor finish and the heating installation itself.

-->

RS-232 communication scripts

Version 2.0 introduces **communication scripts**, with which the installer independently defines how the module communicates with an external device. This makes it possible to handle practically any equipment controlled over RS-232, for example a projector or an AV receiver.



A script can be created in two views, switched at the top of the editor:

- **Flow view:** building the logic from ready-made blocks, without having to write code. It allows simple control commands to be handled quickly.
- **Script view:** writing the communication logic in a **DSL with C-like syntax**. This option is intended for more complex protocols and for people comfortable with programming.

Above the editor there are the remaining configuration elements: the **UART parameters** (baud rate, data bits, parity, stop bits), **frame detection** (the delimiter and the maximum frame length) and the table of **variables** the script operates on.

Regardless of the chosen view, the script describes the full course of the communication: the form of the commands sent, the way the device's responses are interpreted, and how they are linked to Leafs. Thanks to this the external device can be both controlled from within conditions and report its own state.

> Communication scripts are also planned for the **RS-485** bus and for further modules.

Child devices on RS-485

Support for the RS-485 bus has been based on the **child modules** mechanism described earlier. Each device polled over the Modbus protocol is added as a separate **child** of the parent module and has only its own Leafs.

The most significant functional change is **individual transmission parameters**. In previous versions the communication settings applied jointly to the whole bus, which forced the selection of devices working with an identical configuration. Now each child has its own set of parameters, so devices from different manufacturers, with different transmission requirements, can work on a single bus.

In addition, separating the devices into individual child modules organises the functionalities list. The Leafs of the individual meters, inverters or controllers do not mix with each other, which simplifies finding them while building logic.

Screen configuration for M-DOT-M6 and M-DOT-M18 panels

The configuration of the displays in the touch panels has been completely redefined. The rigid layout of icons and text rows has been replaced by a **widget system**, which gives full freedom in designing the interface.

What does the new **Screen Creator** change?

- **Full freedom of building:** Screens are created by selecting a ready-made template and assigning particular Leafs to the widgets, or by adding individual elements manually. If the standard layouts do not meet your needs, each widget can be added separately, building a unique view from scratch.
- **Intuitive arrangement (drag and drop):** After adding widgets to the screen, their position can be changed freely using **drag and drop**. This gives an immediate preview of the final appearance of the screen already at the configuration stage.
- **Element library:** A variety of widget types is available: from buttons and sliders to climate controls and clocks. Widgets come in different sizes, which allows the space on the display to be used optimally.
- **A growing base of templates:** The templates currently available form a base for quick configuration. The system is continuously being extended with new layouts, to speed up work with repetitive elements of an installation even further.